Softwaredokumentation zum Programm

„PhysX KugelLineal“

Projekt Kugel Lineal

Auftraggeber: Ministerium für Kultus und Sport

Entwickler: Team Golden Girls

Stand: 29.11.2015

Inhalt

[Einleitung 3](#_Toc436584958)

[Überblick 4](#_Toc436584959)

[Start des Programms und die Klasse Main 5](#_Toc436584960)

[Das Paket Physics 6](#_Toc436584961)

[Die Methode checkShapeCollisions() 6](#_Toc436584962)

[Das Hindernis ist vom Typ „Goal“ 6](#_Toc436584963)

[Das Hindernis ist vom Typ „Line“ 7](#_Toc436584964)

[Nach der Kollisionsprüfung 8](#_Toc436584965)

# Einleitung

Dies ist die Softwaredokumentation zum Programm „PhysX KugelLineal“,welches im Auftrag des von der Firma „Team Golden Girls“ entwickelt wurde. Das Programm soll das Schießen mit einem Radierer-Lineal-Katapult simulieren. Anforderungen waren mehrere Level und Schwierigkeitsgrade, verschiedene Hindernisse sowie Geräusche bei Kollision mit Hindernissen. Außerdem sollte das Verhalten der geschossenen Kugel physikalischen Gesetzen folgen. Dieses Dokument soll das Verstehen der Funktionsweise des Programms erleichtern, um die Wartung oder Erweiterung zu vereinfachen. Hierzu werden alle Aspekte im groben in ihrer Funktionsweise beleuchtet und schwierigere Teile anhand des Quellcodes erklärt. Zusätzlich zu diesem Dokument ist der Quellcode mit Kommentaren versehen, um das Einarbeiten zu erleichtern. Da das Programm in Java geschrieben ist und JavaFX nutzt, sind Kenntnisse in diesen Bereichen empfehlenswert, um das Programm zu verstehen.

# Überblick

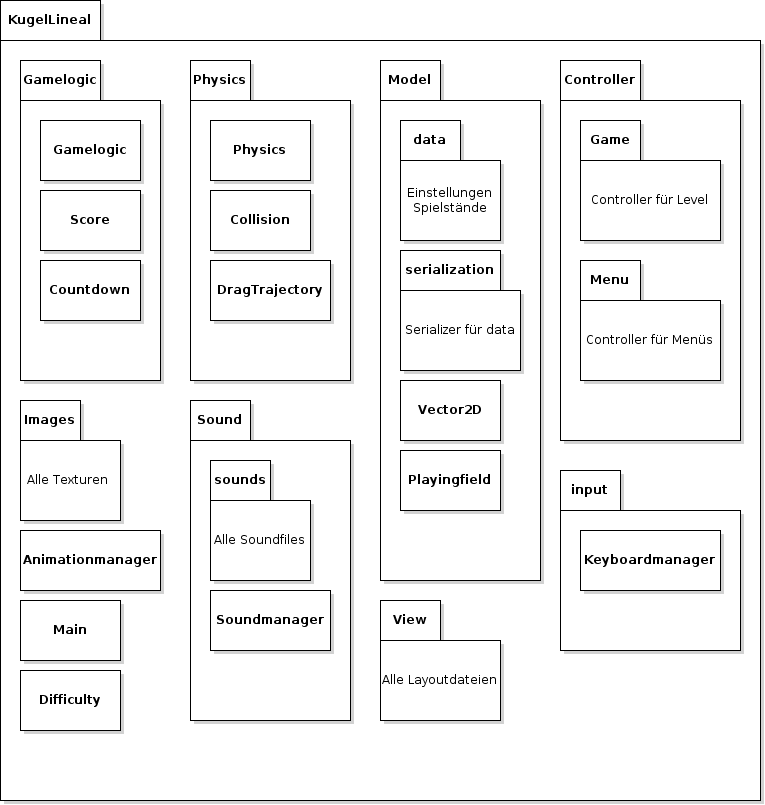


Abbildung Grundlegender Aufbau

Das Programm unterteilt sich in mehrere Pakete:

1. Gamelogic, zuständig für die Rahmenbedingungen eines Levels
2. Physics, zuständig für das Bewegen der Kugel und Kollisionen
3. Model, zuständig für den Zugriff auf Daten sowie das Speichern
4. Input, zuständig für die Erkennung von Benutzereingaben
5. Controller, zuständig für die Buttons etc. auf jedem Screen
6. View, hier werden die Layouts der einzelnen Screens/Szenen definiert
7. Sound, zuständig für das Abspielen von Tönen
8. Images, hier werden die Texturen gespeichert

Für das grundlegende Spielen an sich sind die Pakete Physics und Gamelogic von zentraler Bedeutung. In diesen Paketen werden die meisten Berechnungen während des tatsächlichen Spielens vorgenommen.

# Start des Programms und die Klasse Main

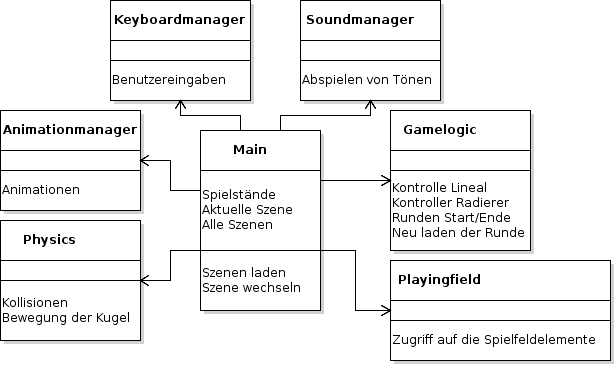


Abbildung Die Klasse Main und ihre grundlegenden Funktionen

Beim Start des Programmes wird die Methode start(Stage primaryStage) der Klasse Main aufgerufen. Die Klasse Main dient als zentrale Anlaufstelle des Programms. Sie bietet Zugriff auf die wichtigsten Klassen (siehe Abb. 2). Außerdem dient die Klasse Main als „Stage“. Auf eine Stage können Scenes angezeigt werden. Die Stage ist im Grunde der Fernseher und die Scenes sind die verschiedenen Sender. Die Klasse Main lädt alle verfügbaren Scenen und startet mit der Scene „MainMenu“, welche das Hauptmenü darstellt. Außerdem werden alle verfügbaren Spielstände geladen, damit diese in „LoadScreen“, also in der „Spiel-Laden“-Scene, zur Verfügung stehen. Alle Scenen sind im Ordner „View“ zu finden.

Von zentraler Bedeutung in der Klasse Main ist die Methode fillScenesMap(List<String> files), welche alle Scenen aus dem Ordner View lädt. Falls eine „Level“-Scene geladen wird, wird zusätzlich noch eine weitere „BaseGame“-Scene hinzugefügt. Dadurch gibt es zu jedem Level „x“ auch ein „BaseGamex“.In der „BaseGame“-Scene sind UI-Elemente, die bei jedem Level gleich sind, zum Beispiel die Position des Lineal-Radierer-Katapults oder die Punkteanzeige. Zu jedem Level gibt es im Ordner /Controller/Game/ außerdem noch einen Controller, welcher die im BaseGame spezifizierten Objekte lädt.

# Das Paket Physics

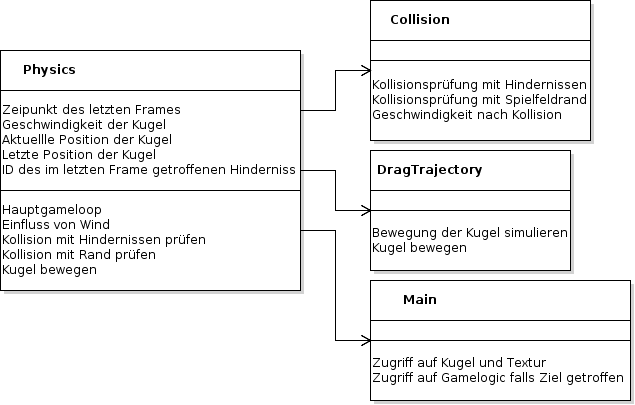


Abbildung Die in "Physics" enthaltenen Klassen und ihre grundlegenden Funktionen

Die in „Physics“ enthaltenen Klassen befassen sich mit dem realitätsnahen Bewegen der Kugel. Die Klasse Physics erbt von der JavaFX-Klasse AnimationTimer und enthält den „Hauptgameloop“in der Methode handle( ), welche jedes Frame aufgerufen wird und folgendes erledigt:

1. Berechnet die Zeit seit dem letzten Frame
2. Simuliert das Bewegen der Kugel
3. Simuliert den Einfluss von Wind auf die Kugel
4. Prüft auf Kollisionen mit Hindernissen, dem Ziel oder dem Lineal
5. Prüft auf Kollision mit dem Spielfeldrand

## Die Methode checkShapeCollisions()

Die Methode checkShapeCollisions() prüft bei jedem Element, ob es zu einer Kollision mit der Kugel kommt. Hierbei gibt es mehrere Fälle zu betrachten.

### Das Hindernis ist vom Typ „Goal“

Falls es zu einer Kollision mit einem Hinderniss vom Typ „Goal“ kommt, so ist der Level erfolgreich abgeschlossen.

### Das Hindernis ist vom Typ „Line“

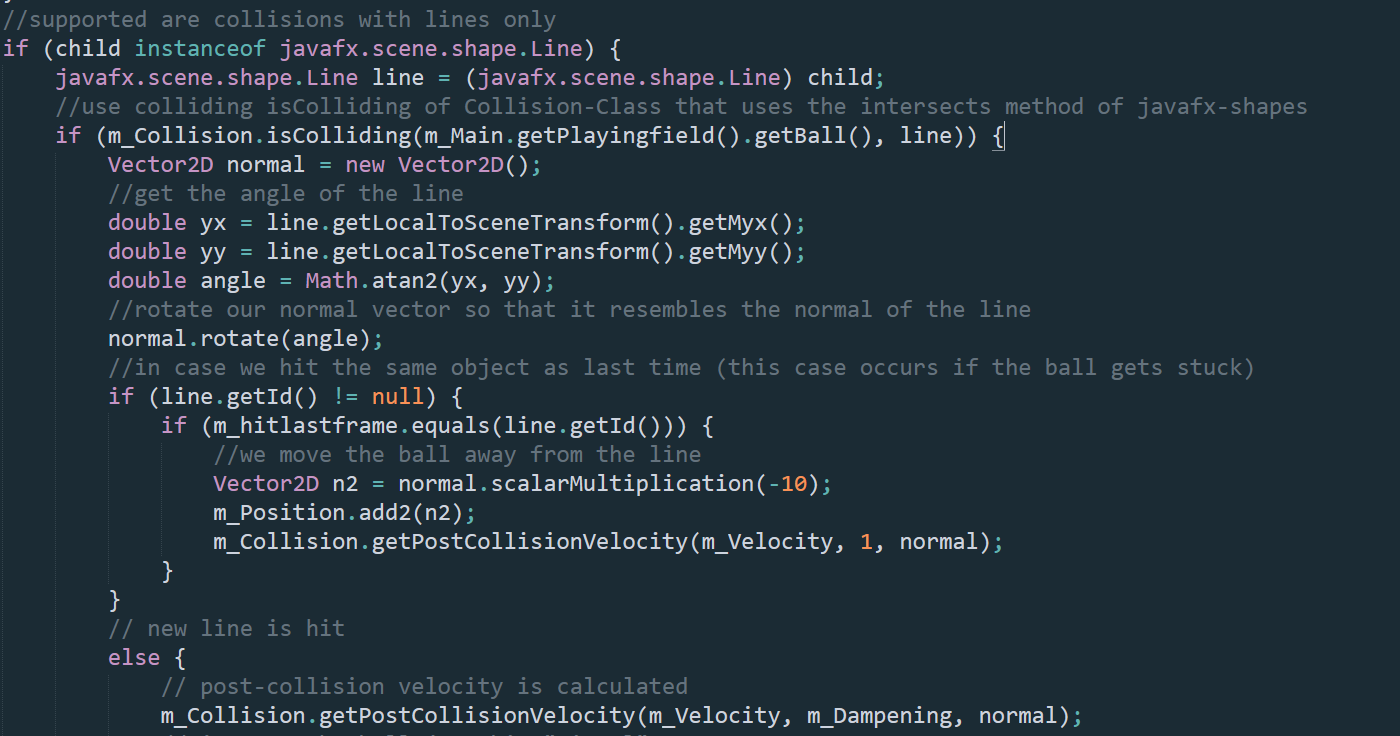


Abbildung Kollision mit einer Linie

„Line“ ist eine JavaFX Klasse, die um die Texturen gelegt wird. Falls es zu einer Kollision mit einer „Line“ kommt, so kann auf 3 verschiedene Arten reagiert werden

1. Falls die Line im letzten Frame bereits getroffen wurde, wird die Kugel ein Stück vom Kontaktpunkt entfernt, bevor die neue Geschwindigkeit berechnet wird. Dadurch wird verhindert, dass die Kugel stecken bleibt.
2. Falls eine Line getroffen wird, die nicht im letzten Frame bereits getroffen wurde, so wird direkt die neue Geschwindigkeit berechnet.
3. Falls die getroffene Line das Lineal ist, so wird geprüft, ob der Ball gerade geschossen werden soll, oder einfach nur im Verlauf des Levels wieder an das Lineal prallt. Falls die Kugel gerade geschossen werden soll, so wird ihr noch zusätzlich die Kraft, die das Lineal auf sie ausübt, angerechnet. Falls die Kugel nicht gerade geschossen werden soll wird die Kollision wie eine normale Kollision mit einer Line gehandhabt.



Abbildung Kollision mit dem Lineal

### Nach der Kollisionsprüfung

Falls es zu einer Kollision kam, so wird der Ball auf seine Position aus dem vorherigen Frame zurückgesetzt und ein zufälliger Ton wird abgespielt. Bei einer Kollision mit einer Line wird zusätzlich die ID der getroffenen Line bis zum nächsten Frame gespeichert, außerdem wird collisionHappened auf „true“ gesetzt.

Die Klasse DragTrajectory ist für die Berechnung der Flugbahn zuständig und das Bewegen der Kugel entlang dieser. Die Berechnung der neuen Position erfolgt inkrementell bei jedem Frame.